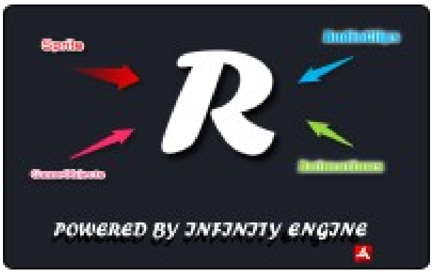


ISI Resource Documentation

Mamadou Cisse

Thanks for the purchase

What is it?

A solution to manages resources placed in unity assets folder like Android R class.
You can access directly to many types of object placed in your project folder on script without register your object in unity inspector thanks to the class R.

We are always open to hearing new ideas for improvements or suggestions and of any problems that you might encounter while using ISI Resource plugin.
You can email us any time at mciissee@gmail.com and we will respond shortly.

How it works ?

ISI Resource provides a set of classes that allow storing, managing and editing the resources of any type specified in Res enumeration.
Also it provides an editor window to organize and choose only the type of the resources that you want to manages.

This plugin really behaves like the R-class of Android, except that unlike the Android, it does not associate the entire identifiers to the resources when generating the R class but it is a static reference to the resources created. This means that there is no need for special functions like **findViewById**.

The plugin is **very fast and fully optimized**.

What's in the Package?

- ISIResource : The core of the plugin.

- ISIResourceEditor : The editor class of the plugin.

- PoolManager : A powerful GameObject manager class.

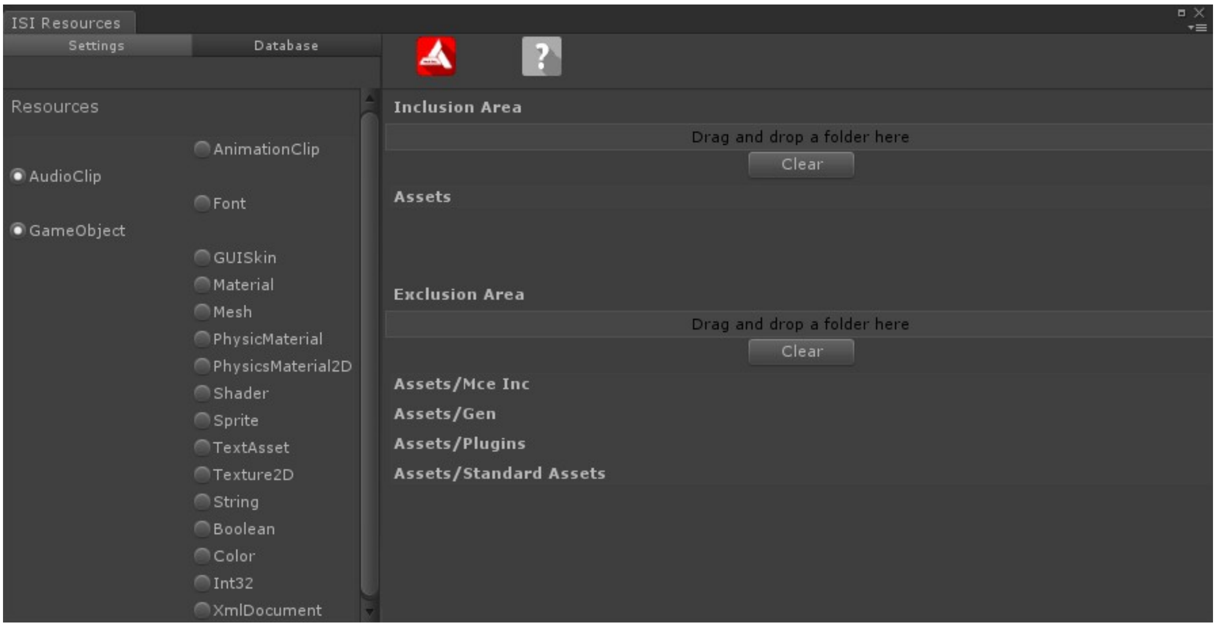
NEVER MODIFY OR DELETE THES ASSETS OR ANY CONTENT PLACED IN THE FOLDER 'Assets/InginityEngine/Gen'.

Getting Stated

Go to the tab **'Tools/InfinityEngine/ISI Resource/Editor'** to open the editor.

The interface is separated in two tab :

- **Setting** : This tab allow you to choose the resources to include in the database and the folders where to find the resources.



Resources

○ AnimationClip
● AudioClip
○ Font
● GameObject
○ GUISkin
○ Material
○ Mesh
○ PhysicMaterial
○ PhysicsMaterial2D
○ Shader
○ Sprite
○ TextAsset
○ Texture2D
○ String
○ Boolean
○ Color
○ Int32
○ XmlDocument

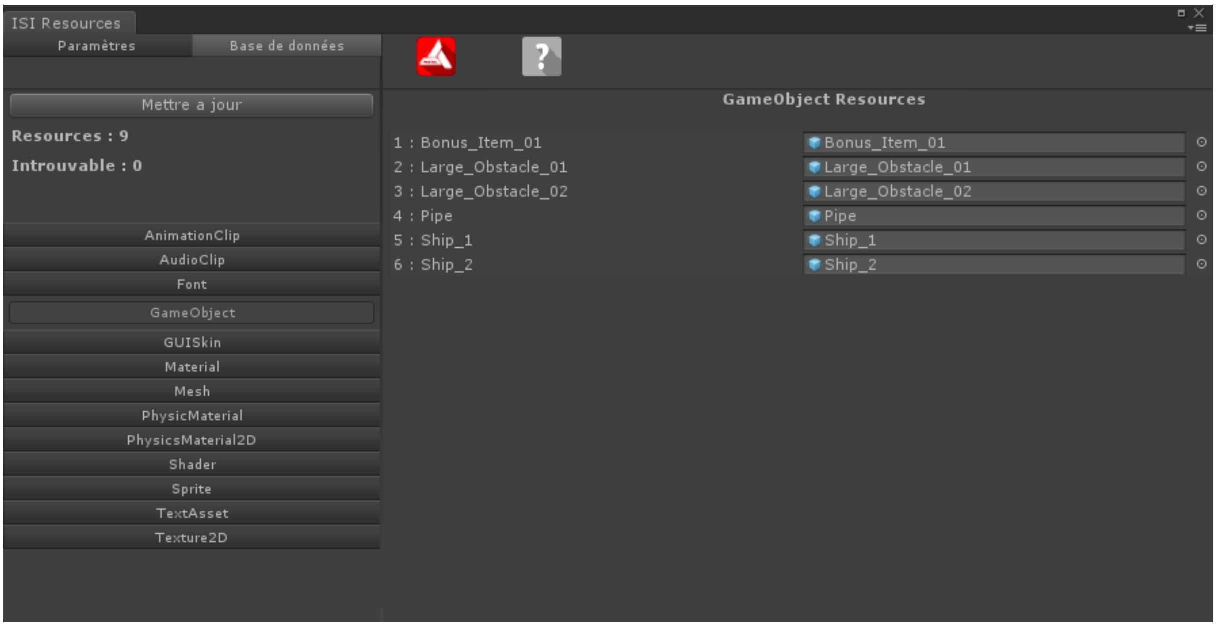
Inclusion Area

Assets

Exclusion Area

Assets/Mce Inc
Assets/Gen
Assets/Plugins
Assets/Standard Assets

– **Database** : This tab allow you to shows the contents of the database.

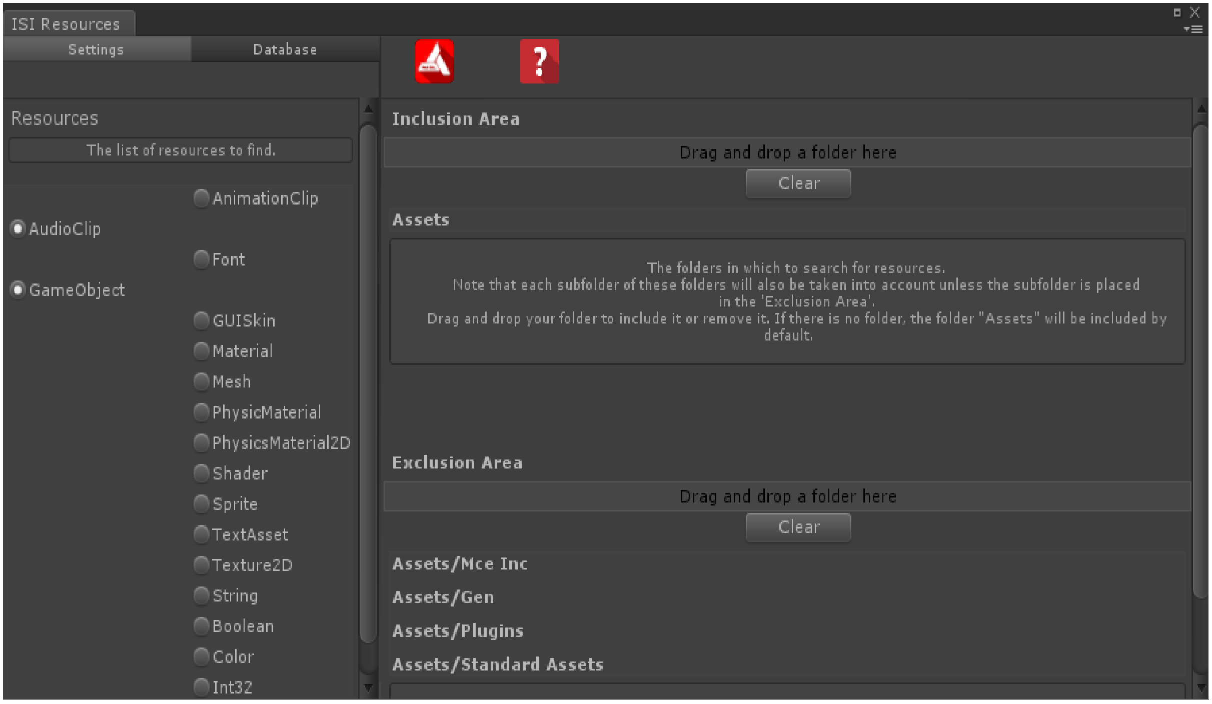


Resources : 9

Introuvable : 0

1 : Bonus_Item_01	Bonus_Item_01
2 : Large_Obstacle_01	Large_Obstacle_01
3 : Large_Obstacle_02	Large_Obstacle_02
4 : Pipe	Pipe
5 : Ship_1	Ship_1
6 : Ship_2	Ship_2

The editor window is localized, you can display a help in the language of your choice (other languages will be added in the future updated)



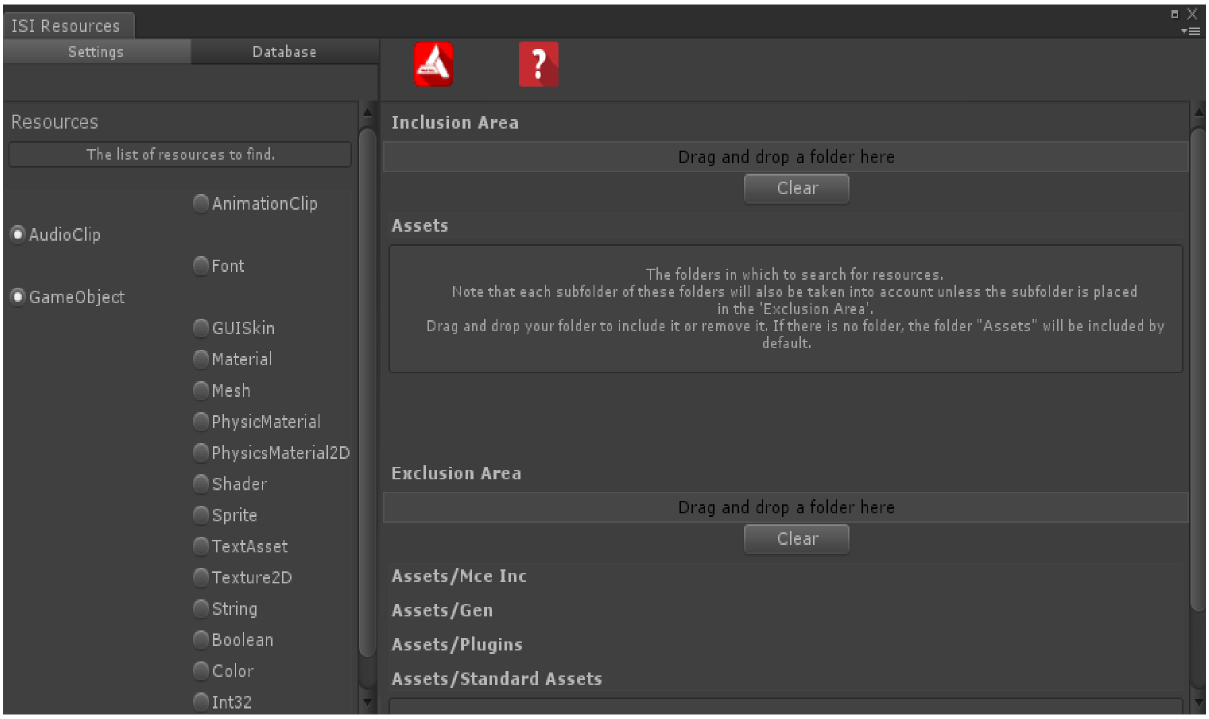
The list of resources to find.

○ AnimationClip
● AudioClip
○ Font
● GameObject
○ GUISkin
○ Material
○ Mesh
○ PhysicMaterial
○ PhysicsMaterial2D
○ Shader
○ Sprite
○ TextAsset
○ Texture2D
○ String
○ Boolean
○ Color
○ Int32

Assets

The folders in which to search for resources.
Note that each subfolder of these folders will also be taken into account unless the subfolder is placed in the 'Exclusion Area'.
Drag and drop your folder to include it or remove it. If there is no folder, the folder "Assets" will be included by default.

Assets/Mce Inc
Assets/Gen
Assets/Plugins
Assets/Standard Assets

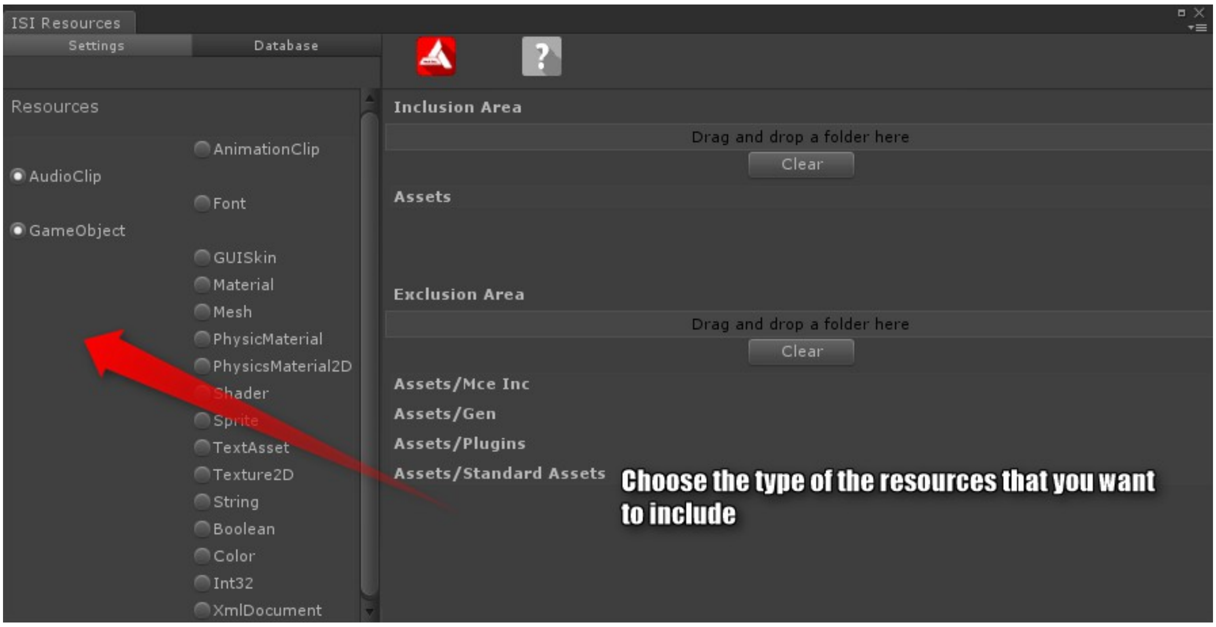

ISI Resources
Settings
Database
Resources
The list of resources to find.
Inclusion Area
Drag and drop a folder here
Clear
Assets
Exclusion Area
Drag and drop a folder here
Clear

○ AnimationClip
● AudioClip
○ Font
● GameObject
○ GUISkin
○ Material
○ Mesh
○ PhysicMaterial
○ PhysicsMaterial2D
○ Shader
○ Sprite
○ TextAsset
○ Texture2D
○ String
○ Boolean
○ Color
○ Int32

The folders in which to search for resources.
Note that each subfolder of these folders will also be taken into account unless the subfolder is placed in the 'Exclusion Area'.
Drag and drop your folder to include it or remove it. If there is no folder, the folder "Assets" will be included by default.

Assets/Mce Inc
Assets/Gen
Assets/Plugins
Assets/Standard Assets

If you open the editor for the first time, there is not a resource in the database, you must choose the type of the resources that you want to add to the database in the left area of the tab 'Setting'.



○ AnimationClip
● AudioClip
○ Font
● GameObject
○ GUISkin
○ Material
○ Mesh
○ PhysicMaterial
○ PhysicsMaterial2D
○ Shader
○ Sprite
○ TextAsset
○ Texture2D
○ String
○ Boolean
○ Color
○ Int32
○ XmlDocument

Assets/Mce Inc
Assets/Gen
Assets/Plugins
Assets/Standard Assets

Choose the type of the resources that you want to include

After you choose your resources, you have to choose the folders where to find the resources and the folders to not check.

Note that you cannot include the resources placed in the folders **'Assets/InfinityEngine', 'Assets/Plugins' and 'Assets/Standard Assets'** or the resources that have a name which starts and ends with '___'.

If you include a folder, all sub-folders of this last will be also included excepts if the sub-folder is placed in the area 'Exclusion Area'.

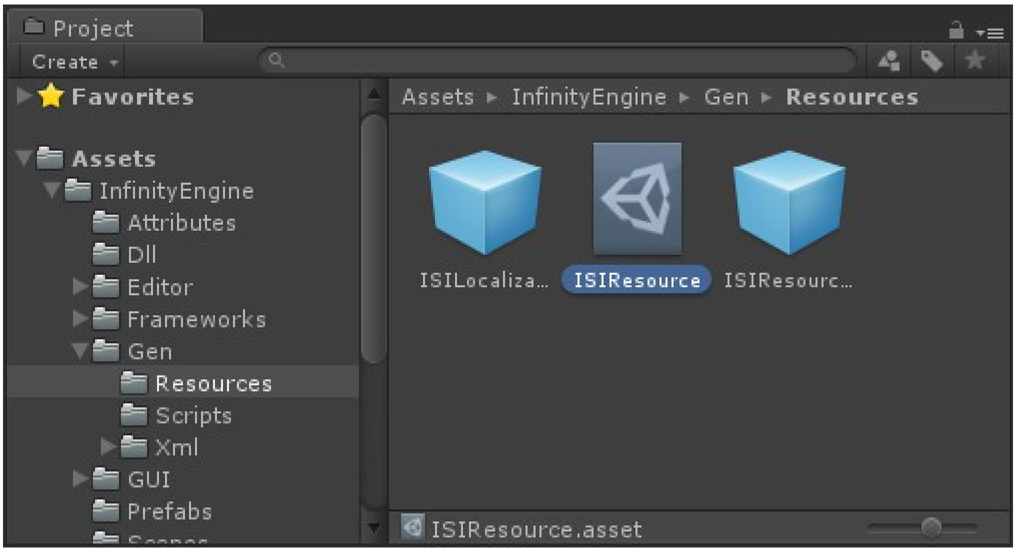
If you exclude a folder, all sub-folders will also be excluded.

Asset Generation

After you configure the project, you can update the database by pressing the keys **'Alt+U'** or by clicking in the button 'Update' of the tab 'Database'.

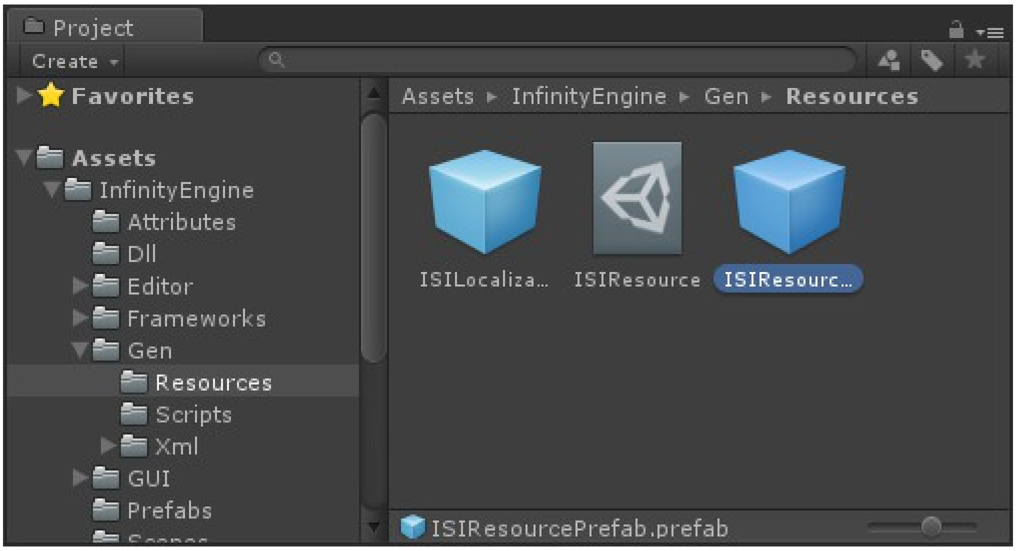
During the update, the plugin search all resources of the types that you choose and creates 3 assets, in the folder **'Assets/InfinityEngine/Gen'**:

1 - ISIResource.asset



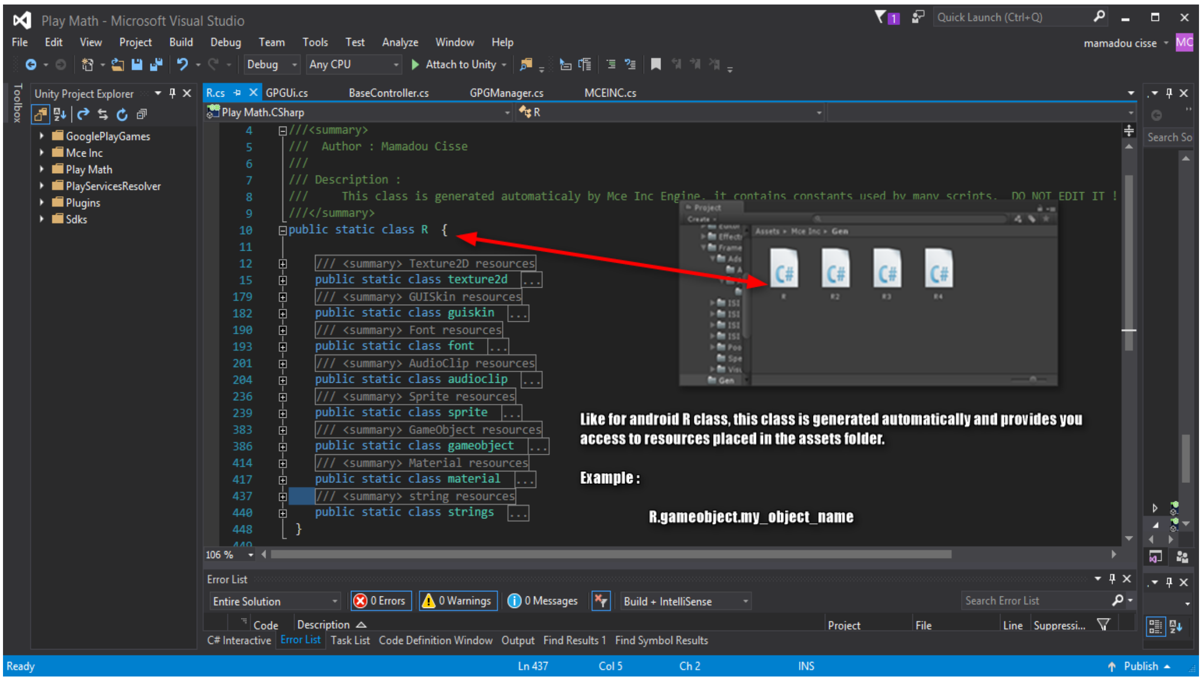
This asset contains all resources and the parameters of the plugin, when you export your project, the resources and the settings are kept.

2 - ISIResourcePrefab.prefab



This is a prefab that will be used at runtime in the build version of your project to provides access to the resource.
In the editor, when you get a resource from the database, it is the ScriptableObject **'ISIResource.asset'** that is used but in the build version in some platforms like Android device, the ScriptableObject is not usable, so the plugin use the prefab to provides you access to the resource. **The plugin has not been tested on all Unity platforms, but it should work in all.**

3 – R.cs



```
///<summary>
///  Author : Mamadou Cisse
///
/// Description :
///     This class is generated automaticaly by Mce Inc Engine, it contains constants used by many scripts.  DO NOT EDIT IT !
///</summary>
public static class R  {

    /// <summary> Texture2D resources
    public static class texture2d ...
    /// <summary> GUISkin resources
    public static class guiskin ...
    /// <summary> Font resources
    public static class font ...
    /// <summary> AudioClip resources
    public static class audioclip ...
    /// <summary> Sprite resources
    public static class sprite ...
    /// <summary> GameObject resources
    public static class gameobject ...
    /// <summary> Material resources
    public static class material ...
    /// <summary> string resources
    public static class strings ...
 }
```

Like for android R class, this class is generated automatically and provides you access to resources placed in the assets folder.

Example :

R.gameobject.my_object_name

Normaly when you want to get a resource from the database, you have to use the method ISIResource.Find<T>(Res, string)
To save you time, the plugin uses the same system as Android, a static class that contains a reference to all the resources.
For example, if you have a GameObject resource in the database which has the name 'myGameObject', you can use the code **R.gameobject.MyGameObject** to get the GameObject.

The fact that this plugin handle a class for you give access to resources makes that you can not name your resources any way.
The name of your resource must respect the same convention as the variable naming in programming, it must starts with a letter or the char '_'

and it cannot contains space or or chars except letters.

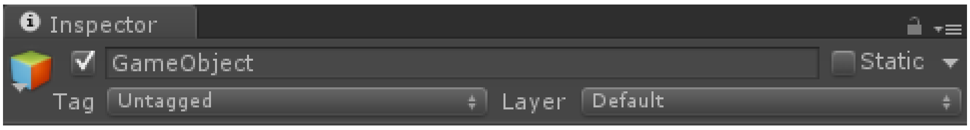
If your resource is misnamed, it will be included anyway in the database but the name of the variable generated in the class R will be different from the name of the resource, all unauthorized chars will be replaced by '_'.
If you have many misnamed resources, the search time of the resources will be increased, so give the good name to your resources.

The class **R** contains a reference to :

- 13 UnityEngine.Object types :
o AnimationClip
o AudioClip
o Font
o GameObject
o GUISkin
o Material
o Mesh
o PhysicMaterial
o PhysicsMaterial2D
o Shader
o Sprite
o TextAsset
o Texture2D

- 5 xml resource types :
o Int32
o Boolean
o String
o Color
o XmlDocument

and the gameobject **tags** and **layers** presents in your projects.



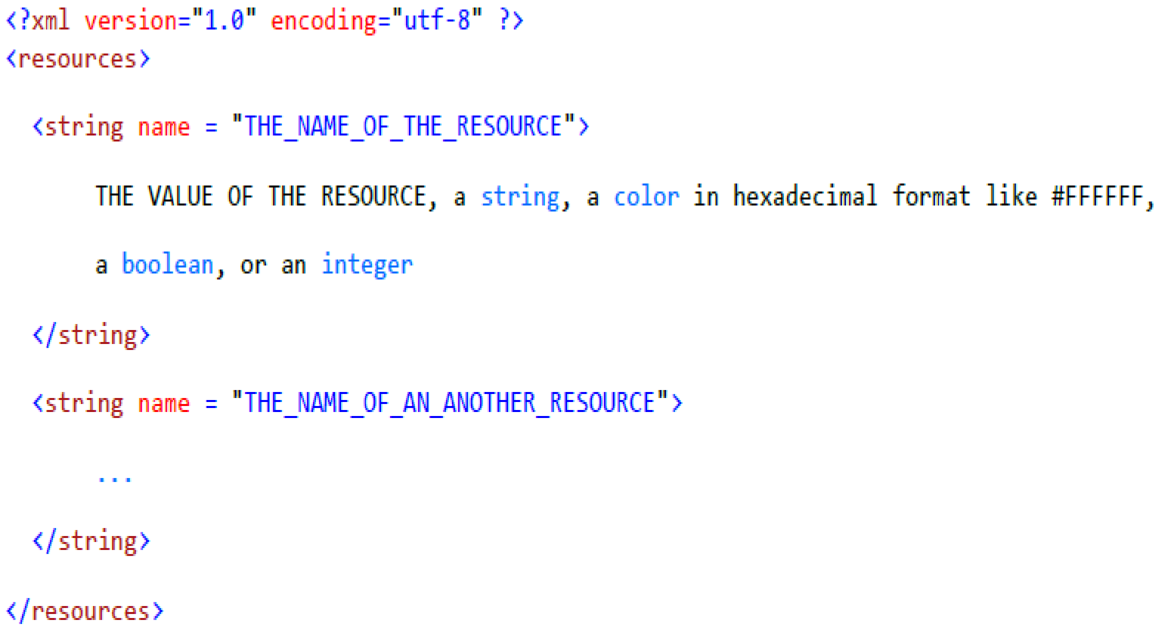
☑ GameObject

Tag Untagged

Layer Default

The resources of type **xml** must be placed in a xml file named as '{0}_res.xml' where {0} is the type of the resource **(String, Color, Int32, Boolean)**.

The type XmlDocument is special, it is a normal xml file. During the research of theresources, if the plugin find a xml file which is not named as {0}_res.xml , it considers it as a XmlDocument resource. The xml resource must look like :



```
<?xml version="1.0" encoding="utf-8" ?>
<resources>

  <string name = "THE_NAME_OF_THE_RESOURCE">

       THE VALUE OF THE RESOURCE, a string, a color in hexadecimal format like #FFFFFF,

       a boolean, or an integer

  </string>

  <string name = "THE_NAME_OF_AN_ANOTHER_RESOURCE">

       ...

  </string>

</resources>
```

You cannot add multiple resource types in the same resource file(for example, the file **'String_res.xml'** must contains only a string resources) and each xml resource type (String, Color, Int32, Boolean) can have only one resource file in the project (you cannot have two file named

'String_res.xml' or **'Color_res.xml'**). If you have a multiple resource files for a xml type, it is the first which is find by the plugin that is used.

How to use ISI Resource?

If you want to use ISIResource functions in script, you must add the following directive in the head of your scripts :

```
    using InfinityEngine.ResourceManagement;
```

There is 2 way to get a resource from the database :

- **Use the static function ISIResource.Find<T>(Res, string)**
- **Use the class R**

The recommended way is the last because the first use a function that take a string parameter and you can pass a bad value in the function and throw an exception.

Example :
In this example, we play an audio clip resource with the name "sound".

```
// the namespace of SoundManager
using InfinityEngine;
// the namespace of R.
using InfinityEngine.ResourceManagement;

public class TestClass : MonoBehaviour{

    void Test{
        //SoundManager is shared with this plugin.
        SoundManager.PlayEffect(R.audioclip.Sound);
    }
}
```

The plugin is shared with the component PoolManager. You can use it to create a pool management system for any GameObject with one line of code or by using the pre-maded component with unity inspector.

```
using InfinityEngine;
using InfinityEngine.ResourceManagement;

public class TestClass : MonoBehaviour{

    void Start(){
        //This simple code create a pool management system for the
        //prefab named "prefab".
        var myGO = PoolManager.Pop(R.gameobject.prefab);
    }
}
```

Final Words

Thanks you if you purchase this asset, if not you can purchase it at
http://u3d.as/J4i

if you like the asset, please rate it.

Support is available :
be emailing mciissee@gmail.com
by Twitter

Make sure you check out our other assets at our assets store page.